RETE 알고리즘을 이용한 계획기반 에이전트 시스템의 실행 성능 향상[1)]

이재호, 김남형°, 송남훈
서울시립대학교 인공지능연구실[2)]
{jaeho, pastime°, uod9642}@ece.uos.ac.kr

Improving Performance of a Plan-Based Agent System Utilizing the RETE Algorithm

Jaeho Lee, NamHyung Kim°, NamHoon Song
Artificial Intelligence Lab. University of Seoul

요 약

최근에 이르러 온톨로지 (Ontology) 및 시맨틱 웹 (Semantic Web) [1] 에 대한 연구가 활발히 진행됨에 따라 이들을 처리할 수 있는 에이전트 시스템의 중요성 또한 부각되고 있다. 이러한 에이전트 시스템은 규칙 (Rule)을 기반으로 하여 해당 조건을 만족시키는 경우에 규칙에 정의된 행동을 실행하도록 설계된다. 하지만 이러한 시스템이 많은 양의 규칙과 조건을 처리하는 경우 특정한 조건에 맞는 규칙을 검사하는 과정에서 많은 오버헤드가 걸리게 된다. 이러한 시스템의 단점을 보완하고자 사용되는 것이 RETE 알고리즘으로서 각 조건과 규칙들을 빠르게 검색할 수 있는 네트워크를 구성하고 특정 조건이 만족되는 경우 이를 참조하는 규칙들의 목록을 즉시 얻어낼 수 있게 된다.

본 논문에서는 JAM 에이전트 시스템에 RETE 알고리즘을 적용시키고 이를 통해 더욱 효율적인 에이전트 시스템을 구성하고자 한다.

1. 서 론

최근의 웹 (Web) 의 발전 방향은 시맨틱 웹 (Semantic Web) [1] 으로 가고 있다. W3C 에서는 이를 위한 온톨로지 (Ontology) 의 표준을 확립하기 위해 OWL (Web Ontology Language) [2] 를 정의하는 등 많은 노력을 기울이고 있다. 이러한 시맨틱 웹의 핵심 기조는 바로 사람이 찾아서 이해하고 활용하는 웹이 아닌 기계가 이해할 수 있는 의미를 부여하자는 것이다.

이러한 시맨틱 웹 환경 하에서 온톨로지를 처리하기 위해서는 특정한 조건을 명시하고 그에 대한 행동을 지정하는 규칙기반 (Rule-based) 에이전트 시스템이 유용하게 사용된다. 또한 이러한 행동을 더욱더 세밀하게 제어하기 위해서는 절차기반 (Procedure-based) 에이전트 시스템의 특성이 요구된다. 이러한 특성들을 모두 만족시키기 위해 고안된 것이 계획기반 (Plan-based) 에이전트 시스템으로서 에이전트의 행동의 기본 양식인 계획을 어떻게 구성하느냐에 따라서 양쪽의 장점을 모두 이용할 수 있게 된다.

본 논문에서는 대표적인 계획기반 에이전트 시스템의 구현인 JAM 에이전트 시스템 [3][4] 의 특징을 살펴보고 이 시스템의 성능을 향상시키기 위해 RETE 알고리즘 [5] 을 적용하여 보다 빠르게 적절한 계획을 선택하여 행동할 수 있는 시스템의 구조를 제시한다.

2. 계획기반 에이전트 시스템

2.1 개 요

계획기반 에이전트 시스템은 에이전트가 자주 처하게 되는 상황과 그에 대한 처리 방법을 미리 정의된 계획으로 표현하고 특정한 상황이 발생된 경우 주어진 상황에 맞는 계획을 선택하여 명시된 절차를 순차적으로 실행해 가는 데에 적합한 범용 에이전트 시스템이다. 이러한 시스템에서는 일반적으로 계획기를 내부 핵심 요소로 채용하고 있어서 상황에 맞는 실행 계획을 수립하고 이에 따라 에이전트가 행동할 수 있도록 한다. 계획기 및 계획 시스템은 계획생성(plan generation)과 계획실행(plan execution)이 완전히 별도의 독립적 프로세스로 분리되어 있다고 가정한다. 즉 계획 생성 단계에서는 계획기가 달성하고자 하는 목표(goal)와 현재상태(current state), 가능한 동작(action)들을 기호논리로 표현한 뒤, 이들을 바탕으로 목표를 달성할 수 있는 동작의 순서를 결정하고, 계획실행 단계에서는 이렇게 생성된 계획을 별도의 실행기에 의해 실행하는 순차적 제어구조로 되어 있다

2.2 JAM 에이전트 시스템

JAM 에이전트 시스템은 JAVA 프로그래밍 언어로 구현된 계획기반 에이전트 시스템으로서 그림 1에서 볼 수

1) 본 연구는 첨단정보기술 연구센터를 통하여 과학 재단의 지원을 받았음
2) 서울시립대학교 전자전기컴퓨터공학부 및 첨단정보기술연구센터 (AITrc)

있듯이 크게 다음과 같은 요소들로 구성된다:

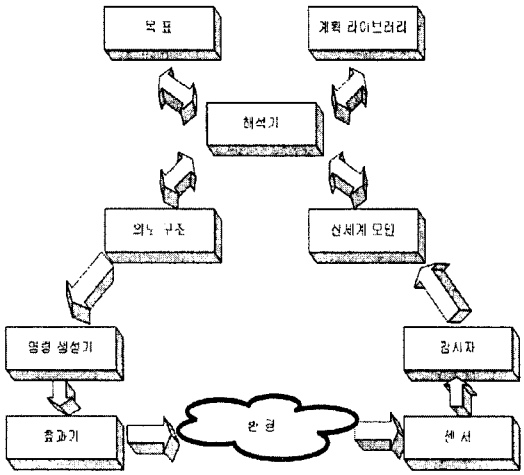

그림 1 JAM 에이전트 시스템의 구조

* 목표 (Goal) : 에이전트가 도달하고자 하는 최종 상태를 표현한다. 각 목표는 자신을 만족시키기 위해 새로운 하위 목표들을 생성하기도 한다.
* 계획 라이브러리 (Plan Library) : 각 상황에 맞는 에이전트의 행동을 명시해 놓은 계획들을 저장하는 데이터베이스이다. 설계자가 계획을 설계하여 에이전트 시스템에게 알려주면 에이전트는 여기에서 자신의 목표 (Goal) 와 주어진 조건에 맞는 계획을 찾게 된다.
* 해석기 (Interpreter) : 에이전트의 뇌에 해당하는 부분이다. 설계자에 의해 주어진 초기 실세계 모델과 최종 목표, 이용 가능한 계획들이 주어지면 해석기는 이를 바탕으로 그에 따른 하위 목표들과 세부적인 계획들을 생성하고 이를 실행하는 역할을 한다.
* 의도 구조 (Intention Structure) : 에이전트의 최종적인 목표를 달성하기 위해서 현재 우선적으로 수행하고 있는 세부 계획의 상태와 그에 대한 하위 목표를 나타낸다.
* 실세계 모델 (World Model) : 현재 에이전트 시스템에서 실세계에 대한 정보 (Fact) 를 저장하는 데이터베이스이다. 에이전트는 실세계 모델내의 정보를 활용하여 행동계획을 결정한다.
* 명령 생성기 (Command Generator) : 의도 구조로부터 에이전트의 상태에 알맞은 명령을 생성하여 실행하도록 한다.
* 감시자 (Observer) : 해석기의 실행 주기 마다 외부계의 변화 및 에이전트의 상태 변화를 감지하는 기능을 수행한다.

3. RETE 알고리즘

RETE 알고리즘 [5] 은 대규모의 규칙들을 포함하는 전문가 시스템 (Expert System) [6] 에서 일어나는 소모적인 매칭 (matching) 연산을 제거할 목적으로 개발되어졌다. 일반적으로 전문가 시스템은 수백 또는 수천가지 이상의 규칙을 포함하는데 각 규칙은 또한 몇 가지(아마도 10가지 이상의) 규칙들을 포함하고 있으며 작업 메모리 (Working Memory) 역시 전형적으로 수백 가지 이상의 조항들을 포함한다. 결과적으로 작업 메모리에 대한 모든 규칙과 조건들을 소모적으로 매칭 하는 데 수만 번의 비교가 필요하다. 이러한 시스템은 계산 시간의 90% 정도가 매칭 연산과 관련되어질 수 있다.

대부분의 전문가 시스템에서 작업 메모리의 내용은 사이클과 사이클 사이에서는 거의 변화가 없다. 이러한 성질을 임시 중복(temporal redundancy)이라고 하는데 이것은 매 사이클마다 소모적 매칭을 불필요하게 만든다. 대신, 이전의 매칭 정보를 저장함으로써 작업 메모리에의 변화된 사항에만 새로이 매칭 연산을 수행하고 나머지는 이전의 정보를 그대로 사용함으로서 연산의 양을 현저히 줄일 수 있게 된다.

또한 각 규칙은 자신에게 필요한 조건들에만 관심을 두기 때문에 이러한 조건들은 각 규칙마다 중복되어 검사될 수 있다. 이렇게 중복되는 조건들을 가지는 규칙들을 하나로 그룹화하면 불필요한 매칭 연산을 더욱 줄이는 것이 가능해 진다. 또한 각 조건은 자신을 필요로 하는 규칙들이 어떠한 것들이 있는지 알 수 있도록 색인화 (indexing) 작업을 하기 때문에 특정 조건이 변화되었을 때 그에 따른 규칙들만을 조사하게 되므로 각 사이클에서 요구되는 연산의 횟수는 상당히 감소된다.

4. 개선된 에이전트 시스템

위에서 설명한 RETE 알고리즘 [5] 을 JAM 에이전트 시스템에 도입하여 JAM 의 해석기가 실세계 모델의 변화에 따른 적절한 계획을 찾아내는 과정에서의 성능 향상을 가져올 수 있도록 시스템을 설계한다.

RETE 알고리즘을 위한 코드를 생성하기 위해 com.irs.jam.rete 라는 패키지를 생성하고 그 내부에는 다음과 같은 클래스들을 생성한다: Rete, Node, RootNode, OneInputNode, TwoInputNode, TerminalNode.

Rete 클래스는 전체적인 네트워크를 구성하여 주어진 조건에 영향을 받는 규칙들을 재빨리 선별해 낼 수 있으며 새로운 규칙을 추가하거나 기존의 규칙을 삭제하는 경우 네트워크를 재구성하는 역할을 한다. 이러한 일들을 가능하게 하기 위해 네트워크는 다음과 같은 항목들을 중점으로 구성하도록 한다:

* 캐시 (Cache) 유지: 조건 비교 연산의 결과를 저장해 두어서 아무런 변화가 없는 상황에서도 다시 연산을 수행하는 일이 발생하지 않도록 한다.
* 동일한 조건 항목의 그룹화: 서로 다른 규칙들이 동일한 조건에 영향을 받는 경우 오직 한번의 연산을 통해 이들 규칙들의 조건에 모두 반영되도록 한다.
* 규칙의 색인화: 특정한 조건에 영향을 받는 규칙들을 바로 찾아서 검사할 수 있도록 한다.

Node 클래스는 실제로 네트워크를 구성하는 각 노드

를 나타내는 객체로서 각 노드는 부모-자식의 관계로 연결된다. 동일한 조건 항목의 그룹화를 위해 하나의 노드는 여러 개의 자식 노드를 가질 수 있다. RootNode 는 조건 부분에 해당하는 노드로서 비교 연산의 결과를 캐시에 유지한다. TerminalNode 는 규칙 부분에 해당하는 노드로서 RootNode 로부터 시작된 네트워크의 가장 하위에 위치하고 이 노드가 활성화 된 경우 JAM 해석기에 해당하는 계획을 이용할 수 있도록 한다. RootNode 와 TerminalNode 는 그 조건의 개수에 따라 OneInputNode 와 TwoInputNode 의 배합을 통해 연결된다.

이를 위해 JAM 의 계획들은 RETE 알고리즘이 인식할 수 있는 형태인 규칙으로 변경되어야 한다. JAM 의 계획이 가지는 precondition 과 context 부분 중에서 실세계 모델과 관련된 조건들로 네트워크를 구성하는 조건으로서 RootNode를 생성한다. 규칙의 경우에는 Terminal-Node 가 활성화 된 경우 그에 해당하는 계획을 JAM 해석기에 알려주도록 하여 조건이 만족되는 경우 해석기가 실행 가능한 계획의 목록에 추가할 수 있도록 한다. 해석기는 그 외의 조건 사항들까지 모두 고려하여 최종적인 실행 가능한 계획의 목록을 얻어낼 수 있다. 완성된 시스템의 구조는 그림 2 와 같이 나타낼 수 있다.

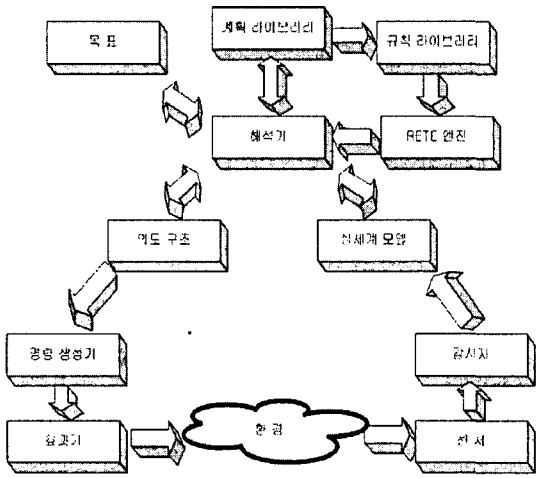

그림 2 개선된 JAM 에이전트 시스템의 구조

5. 결 론

에이전트 시스템의 요구가 늘어감에 따라 다양한 작업을 효율적으로 처리할 수 있는 계획기반 에이전트 시스템의 중요성 또한 커져가고 있다. 또한 시맨틱 웹 [1] 과 같은 방대한 양의 데이터와 그에 따른 규칙들을 처리하기 위해서는 이러한 에이전트 시스템의 성능 또한 크게 개선되어져야 한다.

본 논문에서는 계획기반 에이전트 시스템인 JAM 에이전트 시스템 [3][4] 에 전문가 시스템 [6] 에서 사용되었던 RETE 알고리즘 [5] 을 도입한 개선된 시스템을 제안하여 조건의 변화에 따른 규칙의 선택에 대한 성능향상을 꾀하였다. 이는 많은 양의 조건과 규칙들을 다루는 시스템에 있어서 대대부분의 연산이 이루어지는 부분이므로 전체적인 시스템의 성능 또한 대폭 향상될 것으로 기대된다.

6. 참고문헌

[1] J. Hendler, T. Berners-Lee and E. Miller, Integrating Applications on the Semantic Web, Journal of the Institute of Electrical Engineers of Japan, Vol 122(10), p. 676-680, 2002

[2] Smith, Welty, McGuinness, eds., OWL Web Ontology Language Guide, W3C Recommandation, http://www.w3.org/TR/owl-guide/, 2004

[3] Marcus J. Huber, JAM: A BDI-theoretic Mobile Agent Architecture, AgentLink News, Issue 5, p. 2-5, 2000

[4] 이재호, 박인준, "에이전트 기반 지능형 게임 캐릭터 구현에 관한 연구," 한국 지능정보 시스템학회 2002년 정기 추계학술대회 논문집, 2002

[5] Forgy, C.L., Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, Artificial Intelligence, 19, p. 17-37, 1982

[6] Dan W. Patterson, Introduction to Artificial Intelligence and Expert Systems, Prentice-Hall, p 258-262, 1990